# A Deep Dive on LLM Evaluation

49 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=ISZVCNVIWHK
https://www.youtube.com/watch?v=IsZVCnViwhk

SUMMARY

*The talk focuses on the complexities of evaluating language models (LMs), emphasizing the challenges of scoring their outputs accurately and reproducibly. The speaker, a research scientist at Alther AI, discusses various evaluation methods, including log likelihood, perplexity, and text generation, while highlighting the importance of implementation details and reproducibility in evaluations.*

- *Evaluation of language models is challenging due to the difficulty in scoring natural language responses accurately.*
- *Common evaluation methods include log likelihood, perplexity, and multiple-choice question answering, each with its own strengths and weaknesses.*
- *Log likelihood measures how probable a model's output is based on its training data, while perplexity assesses how well a model fits a given dataset.*
- *Multiple-choice evaluations are cost-effective but may not reflect real-world use cases, as they limit the model's response options.*
- *Implementation details, such as prompt formatting and tokenization, significantly impact evaluation results and reproducibility.*
- *The importance of sharing evaluation code is emphasized to enhance transparency and reproducibility in research.*
- *Downstream evaluations, tailored to specific use cases, are recommended for practical applications of language models.*
- *The speaker encourages the use of established libraries like the LM evaluation harness to facilitate evaluation processes.*

so I'll be giving sort of a very brief uh about 30 minutes there's much more that then can be covered in that time but as much as we can or sort of an opinionated summary um guess an alternate title for this talk could have been basically everything you didn't realize that you needed to ask about LM vals which is sort of a taste of what you should be looking out for um yeah here we go so just yeah a little bit

about me I'm a research scientist at Al Luther AI we're a nonprofit uh research lab uh we you might know us from sort of a number of language models that alther released open source over the years including gptj and GPT newo 20b there were some of the best of the time uh a few years ago things have certainly changed and there are way more options now uh but we also do other research on uh things like interpretability of

models data sets uh distributed training uh evaluation which is this talk uh and we also build and maintain a couple different repositories for uh sort of tooling uh for the open source uh AI ecosystem especially tailored

toward researchers but just useful for practitioners in general uh yeah so I uh in particular I'm a maintainer on the LM evaluation harness uh here's a link uh it's a library that was originally started by some of the founders of

alther AI in 2021 sort of originally just with the uh well scoped goal of sort of onetoone reproducing the evaluations described in uh GPT 3's F shot uh prompting uh in the paper uh on the GPT neom model just to sort of track progress and reproduce these results and since it's grown a lot with the community and uh there's yeah a lot more people working on LMS and evaluation uh and we've been lucky to have it widely

used by a lot of people uh you might know it from the open llm leaderboard uh it's used as the backend there um yeah so uh in this talk uh like I said there's way more than can be covered in just like 30 minutes but I'll try and give like a couple uh deep dives into specific topics I'll briefly give some background on why LM evaluation is so hard and why we need to think so much

about solving many problems here uh I'll give sort of like a very very uh brief uh crash course in how evals are commonly done under the hood some of these uh like gritty uh implementation details that often aren't talked about um and also give some takeaways as a result of this and some sort of minimal best practices and so as a disclaimer for Contex here I am a researcher I don't put LMS into production in my work uh so my recommendations are

going to be somewhat based on my experience in research but at the end I'll sort of touch on which of these points are most applicable still if you're doing uh work in putting llms into production and which sort of don't directly transfer so yeah so there are many reasons that evaluation is hard and also meaningful uh but two that I'll sort of try to focus on here is uh first that scoring is very difficult and later

reproducibility is tough so so what do I mean that scoring is difficult for llm evaluation so there's a very difficult problem which is basically how do we evaluate the mo the responses of a language model in natural language like how the heck do we do this um so like in this in the figure sort of there's a couple different uh clouds one both showing different like potential responses from a language model to sort of a very simple factual question which

is who is the current US president and so in on the left the model answers the correct answer just saying Joe Biden on the right the model says oh well it's definitely not Joe Biden and doesn't really go on to elaborate and so one of these is correct the one on the left and the other isn't um uh but and so like you I if we just look at this question and we have the necessary context we can sort of eyeball

it and say well we know that one of these is correct the other isn't it's fairly obvious to me but uh how would we actually when we want to do evaluation we want to sort of be able to reliably get a measure of our models per performance and do this without much effort and in a way that we can sort of get the same result repeatedly if we want to rerun this evaluation many many times and so we'll want something more

reliable than just sort of a human eyeballing every data point and so we've got a problem which is well so one solution we might think of is okay well if the model's answer to this question includes the the term Joe Biden then we'll mark it as correct and sort of uh treat that as an approximation and go on with our day but the problem is in this example on the right the phrase Joe Biden appears in the model's answer and

yet it's actually saying oh it's definitely not Joe Biden so like the the meaning of these two responses is entirely different but if we use this sort of naive approach of just checking for the right term and the the output then we would end up sort of scoring uh both of these as correct even though one of them is not and so this this problem as sort of a motivating a motivating challenge of sort of how do we check a

language model's work is going to shape a lot of the approaches to evalu ation because if we could sort of uh no matter what give the correct answer on whether a model's uh free form response were correct then we'd sort of have solved the hallucination problem entirely and uh have the perfect language model already uh able to like evaluate these answers and so yeah so people sort of take many different approaches and different situations to resolve this

issue of actually being able to score uh outputs for correctness but there's no uh sort of perfect solution uh everything is either too costly or maybe unreliable uh or has different sort of failure modes and yeah so so with this in mind as sort of like the the thing the constraint that we have to work under there are a couple different ways that we can try to evaluate a model um but in particular how we can sort of probe

these models to inspect their capabilities or characteristics uh and so there are three different ways that people can often interact with these models or sort of try to measure things from them that all talk through um the first is log likelihoods so yeah so so just as sort of a refresher on language models and the necessary background here uh when we feed a piece of input into our language model what we get out is a logits over a

vocabulary so for each of our language models like for example for gpt2 we've got sort of a space of 50,000 different possible uh tokens that it knows uh and when we run some input uh x0 through xn of words or tokens through gpt2 what we get out is this uh sort of for each different possible next word uh gbt2 will give us like a numerical score uh representing like a logic on this uh token and if we apply stth Max then we

can turn this into a probability distribution uh which will basically tell us that gbt2 thinks that there's a 50% chance that the next word is the or uh like a 25% chance that the next word is dog or 10% of cat Etc and so so sort of this is going to be the basic output from our language model this probability distribution over its sort of guess for what the next word or next token will be uh and when we want to most often use

language models we'll sample text from them and the way that we generate text with uh this is that we just uh we feed in the input we get out sort of its probability distribution over uh what its guess for the next word is uh we pick uh we we we sample from this distribution and pick a possible next word and then we feed in that next word to the model and we get out sort of what

the guess for the next next word is uh and so on until we're satisfied and done uh and so in this way sort of a language model maps and input to a a probability distribution uh but a key detail in how we train most uh Auto regressive large language models is that when when we train gbt2 we'll feed in many many paragraphs all at once is sort of the same sample and at at each next word we

can have the model guess sort of what it thinks the next word would be so at the sort of first sentence of this paragraph We can guess what it thinks is going to end that first sentence at sort of the very end of the paragraph We can guess what the very end word is going to be uh and do this all at once so the model is going to give us many many probability distributions both not just predicting

this uh x subn + one but also X n and x sub n minus one and also x sub one and so on so we basically have information about what the model's guesses would have been even though we also know the correct next token uh throughout this um input that we feed in so what we'll get out of a language model is these logits but it's not just a sort of a vector of the sh of shape vocab size but it's sort

of for each sequence position along the sequence line that we feed in uh we get a logic for each elements in the vocab uh and yeah feel free to pop in with questions if there are questions but otherwise I'll keep going so there is firstly just quickly this is this is great and there is there's a not a question it's a comment just a m and a lot of people have up voted just a massive thank you to Haley

for maintaining LM eval and for all the amazing work coming out of alther AI um and we did just have a question is it better to measure or score evals percent Le generated or over a corpus does this vary between modalities and use cases of capability okay I think I could try and answer that one later on um um I guess I see there's and there's one around CL models as well that might oh yeah yeah okay yeah

so so there's a couple details I think I see in the questions here so I guess one thing is that by by simultaneously I I do literally mean simultaneously sort of so when um when gpt2 processes an input uh the many of the operations like the sort of uh MLP layers uh operate on each token individually so you can do this sort of all at once and then in the attention layers you have something called a

causal Mas which is basically like a triangle of sort of visibility across the inputs so for each like for token 10 in the input it can see all of the previous tokens for token 20 it can see token 19 token 18 all the way back to the beginning but for token two it can only see the previous one token and so using this basically we feed in the entire sequence all at once and it sort of only can see at each point in the

sequence it can only see the previous Parts uh but it can sort of compute everything in parallel this is at both uh inference and training time yeah and oh and then yeah one another question is so Clos Source models do produce a logits matrix but they're often not fully available which is a caveat I'll mention later in the talk um just due to how the apis are implemented and what what information is

made uh public so so so now we yeah we've talked about logits and specifically that you can turn these into probabilities but uh like why why is this uh why is this useful other than just simpling text so one way that it's useful and one sort of me measurement that you can take from a model is if you have some input string X and some output string y uh you might want to know sort of how probable it is for your model to

Output say why where why is the correct answer to some question um and so like so like for a simple sentence the cow jumped over the moon if we say feed in the cow jumped over into our language model maybe we wonder How likely is it that our model gives the correct next two words um versus some other word and so this can be used basically for yeah we'll get into a number of things it can

be used for but um like you could use this to sort of measure How likely the like ground truth Target is uh and so the thing that I mentioned earlier is that you can get these logits for every sequence uh position in parallel and so what this means is that for any length of output string y we're going to be able to not just sort of generate these words one by one and check if they match

up but we can do this in just sort of one pass through the language model feed in just one input and figure out the whole probability of a string y so this is a bit of a down slide but basically if we want to compute the log probability of any string y assuming that we're sort of conditioning the model with an input X we can do this just with one path through the model by first taking the concatenation of our

tokenized X and our tokenized y so just like X followed by y um and then uh pass it through the model and we get these logits and these logits are available to us not just at every position within the sequence X but also within sort of the sequence uh y coming after X and so if we just sum over sort of the proper indices of um the Logics uh what we can do is we can go check

basically okay assuming our model has seen all of the tokens in X How likely is it that for the specific uh like token ID that we know uh the first token of Y is going to be how like how much probability does the model assign to that token and so we can check basically how much probability the model assigns to token zero of Y token one of Y and so on and uh just uh sort of combine those

probabilities so if we're using log probabilities just sum the log probabilities if you're not using log probabilities then you multiply but this gets you sort of like very very small numbers for a long number of tokens and why and so what this basically means is that it's very easy to check sort of the probability of some output string uh condition on an input string from a model it's uh just as simple as one pass through the model uh and you can do

other things like you can also check if Y is sort of the most probable next end token for model to produce still in one call just by checking if each of these like true tokens in y are the the most probable for your model to Output as opposed to just how probable they are and so on so this is sort of like a primitive algorithm we can use to get the probability of some output sequence but then why is this useful uh it's

useful in a common uh use case for evaluation that uh people use very frequently this is used in the majority of the open llm leaderboard tasks for example and an mlu a popular data set um to evaluate your model on a multiple choice question if you have some sort of set of Clos a closed set of answer strings y subi uh which in this case in the case of mlu where it's sort of a four Choice multiple choice question a

standardized test um and the uces are A B C or D what we can do is for each of a b c and d we can use the previous algorithm we discussed to calculate the probability of uh producing a conditioned on the input question x uh the probability of producing B of producing C and producing D and so on and we get basically comparative log probabilities of each of these uh potential choices and then we can see

like in this example a is the most likely so what we're going to say is we're going to pretend that a is the answer produced by our model um and so in this case uh for this specific sample question the model's answer is incorrect but maybe if it had put more weight on the answer D and had a higher chance of outputting d uh when we prompted with the question uh then it would get the answer

correct but so basically multiple choice uh is a very common way to do llm evaluations uh the first the first reason just being because of way way cheaper than doing Generation Um for with your language model if you're trying to generate many many tokens uh for say for like a long Chain of Thought to answer each uh question uh this is like a lot of calls to your language model and many steps that you can't really parallelize versus just sort of

passing four different inputs through your model um and so in practice like multiple choice question answering is a cheap way to do evaluation um another huge benefit of doing multiple choice is that because we've not only said that there's only these four possible answer choices for a model to choose from but also we're only comparing the probabilities of these four choices there's no way for our model to give an invalid response or say abstain from answering and take a fifth

uh incorrect Choice um it'll always pick one and its gu might be wrong but it is going to gu um and in my opinion I think that multiple choice uh question answering uh implemented this way is pretty nice for based language models especially small ones that you're training say from scratch because it's sort of a nice way to not have to deal with these like finicky parsing failures and still get sort of like a nice uh uh

measure of your model's capability uh even when it might not be able to coherently generate long form text um but by the same token um vice versa uh if um uh your small model can generate a multiple choice question answering well for just sort of like a single token ranking uh like we described in the previous slide but can't really produce long form chains of thought then this means that your evaluation isn't really matching up well with a real world use

case of say like using the model as a chat bot and so uh in this sense like it's definitely a step away from sort of Downstream usage uh it's also a disadvantage for some evaluation types that Chain of Thought can't be used especially since models are commonly trained with Chain of Thought um and it's also somewhat misleading as sort of real world scenarios in that you don't necessarily want your model to be just solving standardized tests all

day you want sort

of have it handle open-ended questions where it's not given four choices to choose from it's actually supposed to generate and come up with the choice itself uh I guess yeah are there any questions um there are um a few questions comments but we can leave them to the end as well I don't think Mission critical currently cool yeah that sounds good great um yeah so so basically like multiple choice using these log likelihoods is a pretty

common way to evaluate language models but it it definitely has its downsides especially when you're sort of worried more about generation performance or sort of chat usage um yeah and so the second common way to evaluate language models and sort of take a measurement that you could uh sort of assess a model's capability or behavior with is a perplexity so perplexity here's the formula but sort of to describe it in intuitive detail uh we're trying to

measure how well a model fits a given data distribution and the way that we do this is we have some data set which is a collection of documents that are just sequences of words and the the thing that we're going to measure is pretty similar to the the sort of loss we use to train models but um here we take the uh log probability that the model assigns to sort of the true next token for every possible next token that

exists in the data set um so for every token in this data set we check basically uh how How likely the model is to uh output it correctly and we uh average this uh we just this is just average over all of the documents in our data set and over all of the tokens in each document so basically per token how well does the model fit that token and how how well does it predict this data

set or How likely is it to produce this this data um and so this can be done basically very very triv trivially because it's this uh self-supervision where we know the correct label just because we've we've got this input document and we know what the next word is going to be for any sort of prefix in it um so we can take any data set like say Wikipedia or sort of like some Wikipedia page of our choice and just convert it into

something we can measure perplexity on just by checking sort of how well the model fits these sort of true next tokens in the data set and so lexity is a useful tool um especially since it's just basically like using different validation set during training um and you can use it for sort of any data distribution to see how close you're getting but uh it's also not super uh important especially in sort of Downstream use cases for

language models because for an instruction Tunes model or a chatbot um like uh sort of just evaluating how well it fits Wikipedia might be misleading because actually the model is sort of editorializing outputting like a a text maybe perspectives um Etc that wouldn't match with Wikipedia style or like the prompts format might not match for example um however it can still be a useful diagnostic tool especially if you did have sort of a data set or data

distribution that you want your model to be fitting better for Downstream use um yeah so basically perplexity is

a useful tool to have in the toolbox uh it won't be used too too frequently except for sort of like training models from scratch um and so perplexity it seems like a pretty simple approach and it is but there's definitely sort of uh pitfalls and spot guns that uh can occur um for both perplexity and the log likelihood

approach that we discussed before so uh one one complication is that both these log likelihood and perplexity approaches because they're taking sort of either the sum over a number of tokens or averaging over the number of tokens in a data set uh matters what tokenizer you use for your model so if two different models have a different tokenizer the numbers that you're producing might not be directly comparable so a perplexity of a certain value might be easier for a

model with a larger tokenizer to achieve because there are simply fewer tokens to predict over um and so there there are ways to sort of Remedy this um that are uh that can be implemented uh to sort of use the uh tokenizer as part of the system that you're evaluating and have a metric that's normalized with respect to the tokenizer uh but yeah so this is like a lot of text but the important part is basically that there are ways to control

for this but they're all sort of like small implementation details that change what you're measuring and how you're calculating it and then of course the final way that one can evaluate a language model is by generating text from it um this is basically crucially important if we're going to use the model to generate text such as like a chatbot like chat jbt um what we care the most about um and Chain of Thought of course is something

realistic and important for models to use especially in sort of multi-step problems um but there are downsides to doing this sort of generation based uh evaluation which is that again we don't know how to uh like always correctly score free form natural language responses so in the case of multiple choice devaluation we sidestep this by basically saying okay there are only four strings our model is ever allowed to Output in for this document um and so that way

like if there's a string that's not in those four we just disregard it and we only ask the model to predict one of those four strings and we know one of them is correct and the other three are not by construction uh but for text generation the model could output any sort of string and we want to be able to say whether it's correct or not or uh as a matter of degree how correct it is is

it is it half correct is it three quarters correct not at all correct uh so one way that can do this very simply is just sort of uh do a very very rough heris and just say okay I'm going to look for the part in my model's generation where it says the answer is X for some phrase X and then we'll just basically grab what that X is and then check if it matches sort of the gold

standard phrase that we had so like uh as an example like going back to the previous uh sort of comparison of like uh answering who the president is and the answer being Joe Biden we could basically say like tell me what tell me who the president is answer with the answer is X um and then hope that our model follows that format and that it tells us either the answer is Joe Biden or it tells us something else in which

case we know it's incorrect um however this is not great because this just means that we'll be penalizing models that don't comply with our expected format so if we Implement a format of the answer is X models that are trained to produce this format always when it ask the question will do better on our evaluation than models that don't uh and so there's sort of these confounding variables that uh we have to deal with and another way people do this is by

using an llm to basically check if an answer is correct but these are definitely fallible uh and of course there are also other um in this case this is another sort of pain Point caused by tokenization there other reasons uh that sort of generation uh as with other evals can be very finicky so here's sort of like a prompt to the audience uh here's two different prompts these are sort of the first document in human eval

fed into a model um hypothetically is there a difference between these two and if so which do you think is going to give better performance or which one do you prefer so maybe just think about this for like 10 or 15 seconds and then I'll I'll give the answer yeah so so these these two prompts look very very similar but there's one key difference which is these uh two lines at the bottom uh this second prompt ends with one new line a

second new line and then a tab um and so what this means is that if our code generation model has tokens that look like for example a tab and then the return keyword and then uh go on to generate the rest of a solution it means that if our model um uh sort of if the best solution to this function is just a like oneliner return some some expression our our model that's prompted with a tab uh if it tries to generate a

new tab it's going to create a syntax or um so it's forced to generate a token that's like return but without a tab in front of it which might not be a token that it has available or it might be a token that it hasn't really seen during training and so as a result if you evaluate models with uh like one using this prompt with no trailing Whit space at the end and another model with this

trailing wh space human EV valve performance will be something like 5% worse um for a number of models that are tested and so basically this means that like just Bas going down to like the minutest of details in your code that you're uh using to implement your evaluation uh it could drastically affect performance uh and so like if you if you implement your prompts using this trailing wies space and someone else implements it where they they trim off

any trailing white space in the prompt then you'll get different results but it'll be very hard to tell sort of what went wrong uh how things changed so basically in short uh there are a couple different sort of measure options we have available to us all which are trying to overcome this issue of uh not being able to easily score reliably uh free form language model outputs or just natural language in general um and these are sort and the uh

amongst these measurement options there are a number of things that can go wrong or ways you can Implement them subtly not necessarily incorrectly but just differently uh to how is standard um and so on and these Implement details are even implementation details are even not often sort of discussed or mentioned in papers or uh uh Teck reports uh as things you should care about so it's difficult to sort of be aware of them uh a priori

and so all of these challenges

we've discussed are only scratching the surface they're sort of only accounting for did I run my language model correctly like am I getting sort of the correct output from my model they aren't even sort of accounting for external factors like um data set quality of your evaluation like if it's measuring something you really care about if you're overfitting to it and so on and so basically uh scoring models is hard and sort of uh influencing evaluations is difficult and as a result

it's very difficult to achieve reproducibility in evaluations where reproducibility here means basically if uh someone publishes their results they should ideally publish enough details that you could go ahead and sort of write your own code and reproduce the same results they're getting uh within sort of a very small area of margin um and uh so this is key for research or just for sort of iteratively developing better models for production because uh comparisons have to be fair in that uh

advantages aren't given to a new model for example if you've sort of spent much more effort prompt engineering your new uh language model you've just pre pre-trained or your new sort of prototype for production uh if it does better you don't know if this is just because you spent more effort trying to make it good or if the other one if you just prompt engineered for 5 minutes it would also be as good or better um and

so there's definitely a question of sort of what is uh the correct way to set up a fair evaluation comparison and if it is actually holding everything constant or maybe using a prompt format that you know the model was trained with or so on but uh at minimum these evaluation details should be known and should be accounted for so like for example in the table on the right uh the Llama 3 number this is from the Llama 3 released and

these llama 3 70 billion numbers are the ones run by meta because they train the model and this is uh evaluations they ran themselves while developing it but the uh Claude and Gemini results are just results that are self-reported by the model developers in this case Google and anthropic um and they are not as we can see these sort of subtext here they are not necessarily using the same settings across models and in some cases it might not even be super clear what

the Vel opers did if they didn't release the code that they used for these evaluations and so as a result sort of like we see how llama3 measures up against these models but we're not really sure if uh sort of the developers were preferential towards their own model uh or if uh just there were like easier prompts that had more information provided to the model and so on it's difficult to draw like a clean conclusion with these numbers uh so

basically strong reporting standards for evaluation are important but also uh actually reporting enough details in like the tech report that you report uh is very difficult um there are many many things you might forget to account for or just assume that they're standard because it's standard to how your organization does things internally but it turns out that other people don't have that Insight or uh wouldn't implement it in the same way or don't think that's as natural and so basically

in conclusion like I'm going to claim that if you don't see the evaluation code then things are not going to be fully reproducible or at least you're going to have a bad time trying to perfectly reproduce them uh so i' would argue that you should always share your evaluation code if you're doing something like developing a new model or a new method uh in research uh Etc and so yeah so basically reproducibility is important but very

hard and sharing evaluation code especially that's like clean enough for people to read and use can be tricky and that's where libraries like evaluation hardest ands uh other options like Helm or um open compass come in uh by having sort of a easy to reference like gold standard or um just like uh com like frequently vetted uh code based for evaluations uh it's at least possible to sort of not have to worry yourself about all of these tokenization intricacies a

normalization of log likelihoods Etc and more worry about am I doing the evaluation I want to do uh and then it's easier to sort of just say instead of uh implementing all of these evaluations from scratch and saying uh and sharing your code base uh you can instead sort of use these shared code bases uh and uh so yeah we've been lucky to see that like the eval harness at least has been used for a bunch of research on

evaluations has also been used uh when new architectures like Mamba are proposed sort of run on a couple tasks that the community has decided are canonical as like L as log likelihood based evaluations of like BAS language model performance um and so on and then I guess in contrast to other libraries like Helm uh the eval harness we we sort of more intend to just put tools in the hands of practitioners and researchers uh in that sort of many tasks are

supported in our library and it's easy to Define new ones or edit the prompts for your for your own purposes but um it should be up to you which of those tasks you'd like to evaluate on and which are best for you so yeah so so I guess yeah so that was sort of more uh research inflected but as a side note for production there's a distinction that's been made by a number of people I think uh of between model

evals and downstream evals uh where a model eval is more something like the MML U Benchmark uh which is meant to sort of measure how generally capable or generally intelligent with heavy scare quotes uh your uh your language model is to sort of measure it against actual like base model versus a downstream eval which is more I have this concrete use case like maybe as a chatbot for this specific closed domain of answer questions about my

company's documentation uh and I'm going to evaluate my model and see how well it does on this specific task that I want to use it for and nothing else um and so basically whenever possible Downstream evaluations should be used if you have a concrete use case in mind the best eval is one that you've run the code for yourself um instead of in of sort of pulling it from a tech report and just assuming that's how good a model is um

always try to test it yourself um but then even better is an evaluation that you've designed yourself that matches up with your own use case um and if you're trying to measure things that are more subjective like say like sort of how how high quality or how preferred your chatbot is um potentially hiring human evaluators although expensive is worthwhile um and the incentives and sort of uh tradeoff for Downstream evals are very different

because for model

evaluations the thing that we care about is like model quality and we we think of a high MML score as implying that the model is a very good language model in general um and so doing things like training on the test on the train set or the test set um or sort of overfitting repeatedly uh by trying to maximize your MML score over the course of fine tuning uh we think of those as a bad thing

because they're sort of ruining the ability of uh practitioners to draw conclusions based on the evaluation score about how good your model is uh but for a downstream evaluation if the evaluation is literally just how well does the model perform on sort of real world chat transcripts for your actual use case uh it might be beneficial to overfit and try to get to 100% uh on that Downstream evaluation because doing better on that evaluation test just means you're doing

better on exactly the things you care about and nothing else so yeah so uh basically some like high level takeaways in this talk are basically that like implementation Details Matter a lot for llm evaluation models are very very finicky to specific prompts details and formatting to specific sort of implementations of how you um generate text or sort of normalize these measurements um and these can skew not just your your numerical scores that you get out but

also the conclusions you draw about Which models are better than others um and so this matters for research if you're trying to draw Fair comparisons between a new method and old ones um and it also matters for production because like the the sort of tokenization bug that I showed in this uh coding example uh if you were feeding your model A prompts uh in actual production with this trailing white space then all of a sudden your performance would be tanking

and even though the evals looked good on paper if they didn't suffer from the same bug uh you might sort of have a worse model intruction and not know why um and yeah so so this is a very very non-exhaustive list of evaluation wos and things that can go wrong there are many more including data contamination overfitting to your evaluation or having it saturate and no longer be sort of useful for telling models apart uh the

actual measurement validity of your evaluation like is it measuring something that is a real thing that you should care about and is it actually sort of a good match for that if that thing is like capability or something uh and so on yeah so so so in conclusion some of the material in this talk and some of the uh in particular like implementation details are documented in a recent paper uh we put out from alther uh called

Lessons From The Trenches on reproducible evaluation of language models and then also if you're interested in trying to abstract some of this evaluation uh gory detail away then check out the evaluation harness and other tools um we recently launched the ability to wrap model prompts in chat templating thanks to a contribution from hugging face uh we're definitely interested and extending to further uh evaluations Beyond just the sort of text only ones we support uh and yeah we'd

love to sort of hear from you if you're interested in how we can make the library better or if you're interested in helping out even uh since we're very constrained on sort of ability to do all the things that need doing in evaluation um yeah uh in conclusion thanks so much for attending and love to take questions or anything else thanks so much Haley that was that was awesome um I'm um we do have some questions that

I'd love to get to I am interested if people just wanted to start playing around now I know like historically I've gone straight to like hugging face and explored the ml MML data set it's really easy there with the data set View and that and that type of stuff and you can you know get started locally with notebooks and that type of stuff but if people wanted to wanted to get started now um what's the best way to to do that

yeah so we have like an examples folder in the evaluation harness repository that has like a collab notebook where you can just run the scripts um in the harness um and then yeah I'd highly recommend checking out things like the open llm leaderboard and just sort of looking at those data sets and what's in them and thinking about yeah very cool um so we do have a bunch of questions one is um some Benchmark data sets have

some existing errors in them like grammar etc for example hellis swag has 20 6% of samples with error of some sort um does this actually matter in practice yeah I would say definitely I think yeah the hell swag example is particularly egregious and I I like to bring that one up um I so the the place that that's going to come most into effect is say if 10% of your data set samples have errors in them or are

mislabeled um and you're trying to train models that are performing better than 88% like better than 92% % even though there's 10% errors or something and you're sort of fighting over 89 90 91% accuracy you're very very close to sort of the point at which the Benchmark has outlived its usefulness if it hasn't already um sort of any Improvement getting toward 100% accuracy is only going to be a product of sort of overfitting to that evaluation and uh

not a product of the model actually getting better so there's definitely a point in which evals stop becoming useful for telling models apart in capability which is the thing we might care about from them um it's not always what we care about but if that's what we care about like ranking models then yeah that makes sense um we have a question from kit blows my mind we can get all of these so this is back to the

um um getting all the the logs and all of that um get all of these simultaneously but so kids wondering in practice how do we get all of these probabilities um for for one given inference task yeah so or maybe if it's possible to elaborate on the question um so there's two components I guess like um when you evaluate on a given task obviously you're going to have to run inference on like each document

separately um you can definitely batch those documents together but you're going to have to sort of figure out what the model's output on each different input would be separately but in terms of like the predictions at each time step uh uh in in parallel this is sort of a a a characteristic of Transformers as well as other sequence models like ssms but it's it's the it's the reason that when you train a gpt2 model with 20

248 tokens in context you don't have to run 248 steps to get the model's prediction at time step one time step two time step three you can just sort of run your batch through the model a single time and so by construction our models are sort of able to do this um parallel processing and this is why you can only like you can process your entire prompt up to a certain point in only one step or sort of pre-filling

before Generation and it's why you can train without doing sort of a number of steps equal to the number of tokens you're feeding in awesome thanks for that um extra color um we'll get to a few more questions and people who are here if you could upvote the ones that that um you'd like to hear answered as well shamik has a question and you have talked about this a bit what are your thoughts on llm as a judge in that case how to ensure

that the judge model is correct I I want to add a bit more flavor to this question just from you know the way I think about and you you really spoke to this with your Joe Biden example we're actually in a weird situation right where um all our old tools now we have llms generating a lot of natural language all our old tools of you know pattern matching and ner and um all these nlu tools aren't

quite enough given that essentially when llms respond to questions or try to answer things um we do actually want to know something about the semantic meaning of the text they produce not necessarily um the the string matching or regular expressions or anything anything like that so we are in a situation where kind of even if we didn't want to use llm as judges we're kind of there's a forcing function of some sort um so given that context um

what are your general thoughts on llm as as judge Haley yeah so I think there's an interesting tension between sort of the fact that we want to use llms as a judge or sort of human evaluations and annotations as like scorers for tasks that are inherently more difficult or complex um because it's harder to sort of come up with a heuristic that's going to close match the performance like like if it's sort of a more subjective or

just like multi-step um uh reasoning process to like decide whether a task uh has been done done correctly um it's more it's more desirable for us to use an llm as a judge because we want to just sort of be able to have something that spits out a number that says whether it's correct or not but at the same time this is exactly where llm as a judge is going to be uh less potent um because just these models are going to

be better at the simpler task of say extracting like the multiple choice uh sort of answer that was produced from a model's output so like I think LM as a judge is like a very valuable tool but I'd like to see more work done on sort of where they fail and what their limits of capability are because like if you're trying to use gpt3 to evaluate GPT 4 on a task that gpt3 can't do you're likely

going to have a bad time um yeah so in short I think it's a useful tool but um you know more attention has to be paid to sort of what is the performance of the judge model before you use it willy nilly great um someone has asked about um The Arc uh Benchmark um and that it's been offered on kaggle a $1 million prize for the winner why is Arc a harder

Benchmark yeah yeah um so Arc is much more focused to like generalization um many of the evals that we use for language models are ones that um while while it does sort of measure like like if a model can perform the task that means it's capable and useful um at its core many of these like MML U are sort of a combination of okay can the model do in context learning but then just like doesn't know enough facts

and so these are things that if a model has just seen it enough in it's pre-training Corpus because it's listed as a fact on the internet then it's going to be able to answer this question and so like I think for uh Arc and for things that require many many hops of reasoning it requires at minimum a lot more scaffolding around the language model to perform these multiple leaps and so like current benchmarks are often memorization tests um or at least sort

of can be heavily improved on by just increasing memorization whereas sort of performing tasks that require many many weaps of reasoning again reasoning with scare quotes um is a much more difficult challenge right um we have a couple of really nice practical questions around and one is um one's from May the others from Simon Willison um Simon speaking later today everyone if you want to um check out uh a talk that we're all excited about but um they're asking

around just how to get a model to reliably give a multiple Choice answer without like a whole bunch of fluff or not giving the answer as well yeah so I guess one component here which we have not integrated or explored in uh the LM evaluation harness is like structured generation tools that can actually sort of enforce say like we'll do free form generation from a model but it won't be free form because we constrain it to only sort of output the

appropriate tokens um so so like things can be done to sort of mask out other tokens probabilities and sample from the model but prevent it um I guess another component here is like for for the most capable models I guess just prompting it with the system prompt to directly and only return the answer but I would say like if you don't have the ability to do structure generation because you don't have that sort of access with your API

um asking nicely and um like um I think yeah It's Tricky because I think a lot of models are trains nowadays to sort of always produce large chains of thought either because it helps them or just because people like the conversational style so I think while like system prompts can help on this if you don't have access to some way of like sort of more reliably ensuring with structure generation or log likelihoods it's going to be

tricky oh um I think asking nicely is a nice note to end on as well um and particularly given you know the breadth of everything you covered covered here um super grateful um for you sharing all all of your wisdom from from the the bleeding edge of all of this work Haley um and thank you all for joining as as well we didn't get to all the questions um as as is usually the case but um we

can um get to the rest in in in Discord as well so feel free to continue the conversation there um and definitely uh we'll share more of the links that Haley has shared um are we able to share your slides as well Haley or yeah can send a link um and I don't believe I have access to the Discord channels but I can I'm happy to answer questions after the pack amazing oh I'll send the link to

you on a Twitter DM as soon as we jump off the the call um but thank you one once again and thank you all for joining yeah thank you so much for having me and thanks everyone for the questions awesome all right see you soon everyone